Présentation du TD/TL du cours de représentations parcimonieuses

On attend de vous que vous soyez inventifs et autonomes. Décrivez au maximum vos idées et vos astuces. Essayez d'universaliser autant que possible ces astuces pour qu'elles n'en soient plus complètement (sans oublier que l'universalité absolue n'est pas atteignable).

Pensez à faire du code propre, lisible et compréhensible par quelqu'un de complètement neuf sur le problème qui lui arrive soudainement dessus et doit le plus vite possible être capable d'être productif.

Écrivez votre présentation (en prévision de la soutenance) sans attendre. Vous allez ainsi gagner du temps.

Première partie du TL/TD : bancs de filtres

- L'énoncé de l'exercice, fait en cours (et corrigé en cours), dit :
  - On a un signal harmonique composé de la somme de 4 sinus. La fréquence fondamentale est 880 Hz et la fréquence d'échantillonnage est de 8000 Hz.
  - Combien de décompositions successives faut-il pour récupérer un sinus au plus par bande ?
  - À quelles fréquences ont été du coup transposées les 4 sinus initiaux ?
  - Quelle(s) méthode(s) utiliser pour trouver les fréquences de ces 4 sinus ?
- Question 1. Reproduisez en pratique cet exercice (dans le langage que vous préférez : Matlab/Octave/Python). Ça veut dire, pour commencer, faire le son correspondant à ça (le son peut durer environ 1 seconde, ou plus : mais, attention, plus il sera long, plus l'analyse prendra de temps). Puis écrivez le son dans un fichier .wav (avec "audiowrite" pour octave/matlab). Écoutez-le pour vérifier que vous ne vous êtes pas trompés (avec le logiciel "audacity"). Puis analysez ce son comme proposé dans l'exercice.
- Question 2. Ajoutez un vibrato (modulation de fréquence) d'amplitude 20 Hz et de fréquence 5 Hz au signal de départ, et commentez ce qui se passe. Note : prenez soin, en calculant le spectrogramme par exemple (ou le scalogramme), que vous avez bien ajouté le vibrato que vous vouliez ajouter ; il y a un peu d'équations à poser, pour correctement ajouter le bon vibrato. Puis augmentez progressivement l'amplitude du vibrato, et commentez.

Aides :

- Matlab (Octave) : `https://fr.mathworks.com/help/matlab/index.html`
- Python : numpy, matplotlib, scipy.signal semblent utiles ; et voir : `https://www.f-legrand.fr/scidoc/docimg/sciphys/caneurosmart/pysignal/pysignal.html`

Deuxième partie du TL/TD : ondelettes

On va considérer deux applications possibles : le débruitage et la séparation de sources. Commencez par celle que vous voulez.

Débruitage

Je vous propose d'abord de débruiter un signal avec les ondelettes.

- Déjà, pour commencer, essayez de reproduire l'exemple donné en cours.
- Il faut que vous mettiez en place une mesure de la qualité du débruitage obtenu : RSB avant débruitage, et RSB après débruitage.
- Puis, essayez avec plusieurs types d'ondelettes.
- Il peut aussi être intéressant de jouer sur le nombre de décompositions.
- Jouez sur les autres paramètres disponibles (seuil, variance du bruit, etc.), pour constater et mesurer effectivement les effets de chacun d'eux.
- Essayez de régler automatiquement le seuil : le seuil universel de Donoho par exemple est communément utilisé en traitement des images, etc. (il y a des dizaines de seuils automatiques).

Séparation

Puis, je vous propose d'effectuer la séparation de sources avec les ondelettes (note : ici, on se place dans un cas difficile : on a moins de capteurs que de sources, puisqu'on a un capteur et deux sources) :

- Déjà, pour commencer, essayez de reproduire l'exemple donné en cours.
- Il faut qualifier la qualité de la séparation de sources obtenue.
- Puis, essayez avec plusieurs types d'ondelettes.
- Et jouez sur le nombre de décompositions.
- Et jouez sur les autres paramètres disponibles (seuil, fréquences des deux sinus...), pour constater et mesurer effectivement les effets de chacun d'eux.
- Essayez de déterminer automatiquement le seuil

Troisième partie du TL/TD : apprentissage supervisé

Voir les deux répertoires SONS et SONS-VC, où un tas de petits fichiers .wav sont donnés. Chaque répertoire contient trois sous-répertoires : oiseau1, oiseau2, oiseau3. Chacun de ces sous-répertoires contient 1000 fichiers .wav pour SONS et 500 pour SONS-VC. Chaque fichier .wav contient un « trille » de 50 ms.

- Ces chants viennent de trois oiseaux différents : il s'agit de les classer automatiquement, en utilisant des classifieurs comme les KNN et les GMM (puis les SVM éventuellement).
- Les chants sont simples : l'émission d'un signal harmonique sans modulations, et à peu près toujours à la même amplitude, par tranches de 50 ms.
- La fréquence de la fondamentale, pour un oiseau donné, n'a pas toujours exactement la même valeur : il y a une certaine dispersion (petite) autour d'une valeur moyenne ; même chose pour les amplitudes.
- Ce qui varie entre les trois chants, c'est la fréquence moyenne de la fondamentale et son amplitude moyenne.
- Cependant, les trois fréquences moyennes et les trois amplitudes moyennes sont proches l'une de l'autre, ce qui rend la classification un peu difficile.
- Lisez les sons avec "audiowrite" (avec une boucle "for").
- Extrayez deux caractéristiques sur chacune de ces tranches : la fréquence, et l'amplitude de la fondamentale. Ce, avec les méthodes d'analyse spectrale que vous connaissez.
- Possiblement, vous pouvez régler certains paramètres : taille des trames, taille des FFT, fenêtre de pondération utilisée, etc.
- Je vous donnerai éventuellement le code permettant de calculer les SVM (1 classe, 2 classes, etc.).
- Lisez le fichier README que je donnerai avec le code des SVM.
- Vous aurez aussi à régler les paramètres des divers classifieurs.
- Et le but est de me fournir des tableaux du type de ceux présentés dans le cours (slide 303 ; explications slide 304) : c'est ça qui m'intéresse. L'idée est d'utiliser les fichiers de SONS pour faire l'apprentissage des classifieurs, et donc d'avoir une estimation de l'erreur d'apprentissage (ou risque empirique), et pour obtenir les performances de généralisation (ou risque espéré, ou réel) de vos classifieurs ; puis d'utiliser SONS-VC pour obtenir l'erreur de validation croisée.
- Note au sujet des SVM : le programme à modifier pour votre TL est le programme "uncontretous.m". Il est basé sur les SVM 2 classes, et la stratégie "un contre tous" est utilisée (puisqu'il y a trois classes).
- Attention : n'oubliez pas de normaliser la moyenne (à 0) et l'écart-type (à 1) des features extraits.

Quatrième partie du TL/TD : MP/OMP/BP

L'idée, c'est que vous fassiez de la compression audio avec les techniques de "pursuit". La première problématique (la plus importante), c'est bien sûr d'obtenir la parcimonie la meilleure possible (le taux de compression le plus grand) ; cependant, il faut se rappeler que le temps de calcul est important lui aussi (par exemple, pour la téléphonie, il faut rester en temps réel : c'est-à-dire que pour traiter une seconde de son, il y a moins d'une seconde en temps de calculs disponible).

- Comparez les performances de MP, OMP et BP sur un son enregistré par vous
  - dite une petite phrase (de quelques mots), en soignant l'enregistrement (pas de saturation, et utilisation maximum de la dynamique)
  - pas trop longue (2 ou 3 secondes), sinon le temps de calcul peut devenir trop grand (selon l'ordinateur et les techniques utilisés)
- Ce qu'il y a à comparer :
  - les erreurs finales pour les trois types de "pursuit" (les RSB)
  - les temps de calcul pour les trois types de "pursuit"
  - les taux de compression obtenus pour les trois types de "pursuit"
- Attention : utiliser les mêmes "frames" pour les trois techniques, afin de comparer ce qui est comparable
  - essayez d'autres "frames" que Gabor plus Hanning (comme moi j'ai fait)
  - concaténez diverses sortes de "frames" : les TFCT, plusieurs types d'ondelettes, plusieurs "tailles" d'ondelettes, etc.
- Attention à la fréquence d'échantillonnage (44100 Hz n'est pas nécessaire, pour la parole ; 32000 Hz ou même 16000 Hz suffisent)
- Il y a un tas d'autres paramètres à régler/tester : lire les docs des diverses fonctions disponibles
  - par exemple : il y a trois algorithmes pour le calcul du "matching pursuit orthogonal" (OMP), qui ne s'exécutent pas à la même vitesse ; essayez de mesurer ça aussi ; ainsi, regardez la doc de "franamp" : voir 'qr' (QR based method), 'chol' (Cholesky based method), 'cg' (Conjugate Gradient Pursuit), et 'auto' (c'est "franamp" qui se charge de choisir l'algorithme)

Stéphane Rossignol – 2024/2025